Sort Esterno

Leggere sezione 8.3.1 di Riguzzi et al.
Sistemi Informativi

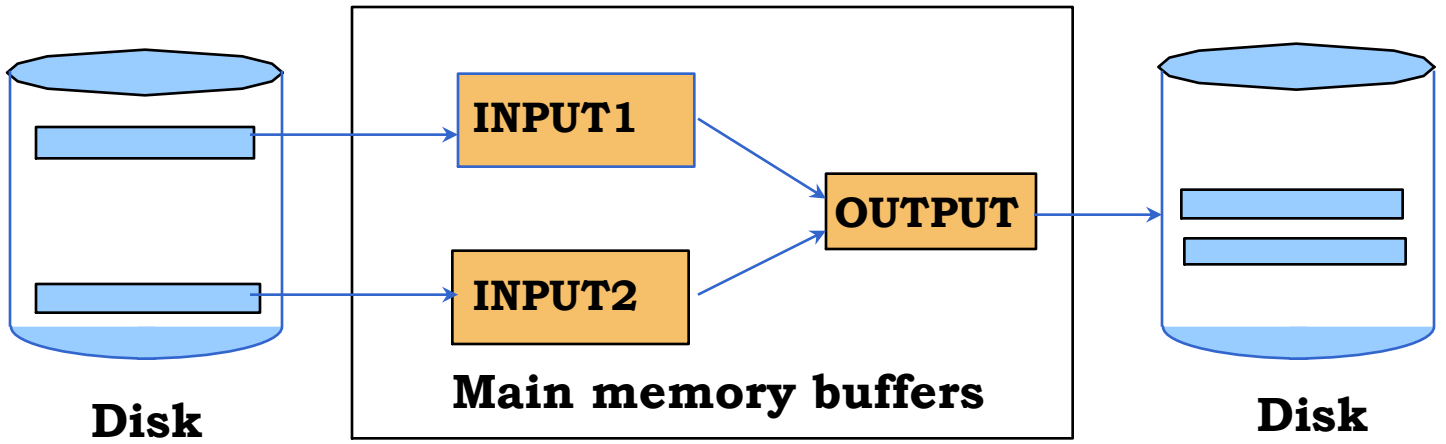
Lucidi derivati da quelli di R. Ramakrishnan e J. Gehrke

Perche' ordinare?

- Dati richiesti in ordine
  - Ad esempio, trova gli studenti in ordine crescente di eta'
- L'ordinamento e' utile per eliminare copie duplicate in una collezione di record.
- L'algoritmo di *Sort-merge* join richiede l'ordinamento.
- Problema: come ordinare 100GB di dati con 1GB di RAM?

2-Way Sort: Richiede 3 Buffers

- Passo 1: Leggi ogni pagina, ordinala in memoria centrale, scrivila.
  - Solo una pagina di buffer e' usata
- Passo 2: leggi due pagine ordinate, fai il merge e scrivi l'output
  - Tre pagine di buffer sono usate



Merge a 2 vie

- Siano p1 e p2 due puntatori: p1 (p2) punta al primo record del blocco INPUT1 (INPUT2)
- Finchè p1 e p2 puntano a record entro INPUT1 e INPUT2
  - Sia p il puntatore tra p1 e p2 che punta al record con chiave più piccola
  - Copia *p in OUTPUT. Se OUTPUT è pieno scrivilo su disco
  - Incrementa p
- Copia in OUTPUT i record da p1 alla fine di INPUT1
- Copia in OUTPUT i record da p2 alla fine di INPUT2

2-Way Sort

- Passo 3: fai il merge di due porzioni di due pagine ciascuna
  - Leggi la prima pagina delle due porzioni, comincia a fare il merge e scrivi il risultato su una terza pagina
  - Quando una delle pagine delle due porzioni finisce, leggi la seconda pagina
  - Tre pagine di buffer sono usate
- Passo 4: etc.

Two-Way Merge Sort esterno

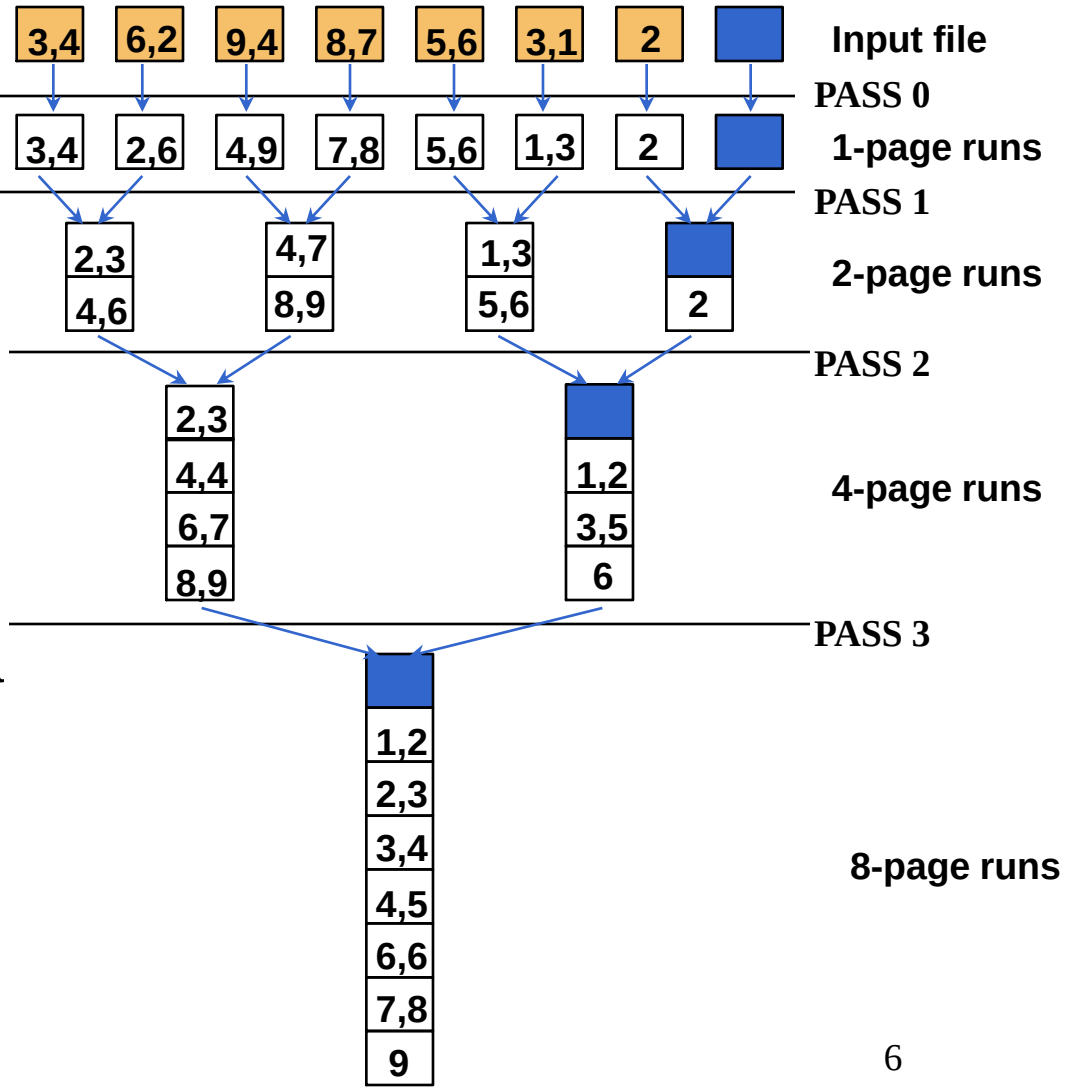
- Ad ogni passo leggiamo e scriviamo ogni pagina del file
- N pagine nel file => il numero di passi e'

$$= \lceil \log_2 N \rceil + 1$$

- Quindi il costo totale e':

$$2N\left(\lceil \log_2 N \rceil + 1\right)$$

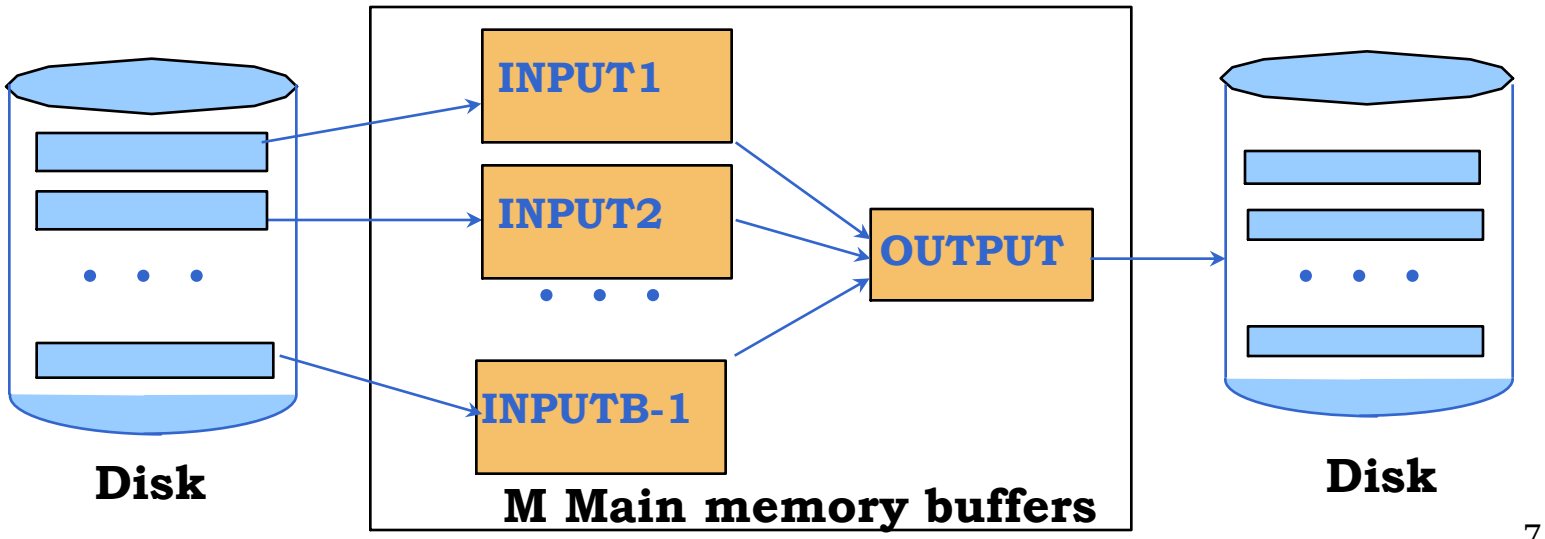
- *Idea:* ***Dividi e conquista*** ordina i sottofile e fondi



Multiway Merge Sort Esterno

✉ *Piu' di 3 pagine di buffer. Come utilizzarle?*

- Per ordinare un file di N pagine usando M pagine di buffer:
  - Passo 1: usa *M* pagine di buffer. Produci $\lceil N/M \rceil$ file di M pagine ciascuno (tranne l'ultimo che puo' averne meno).
  - Passo 2, …, etc.: fondi *M-1* file.



Merge a B-1 vie

- Come quello a 2 vie
- Unica differenza: come p si prende il puntatore al record con chiave minima tra gli M-1 record correnti di ciascun blocco di input

Costo del Merge Sort Esterno

- Numero di passi=1 + $\lceil \log_{M-1} \lceil N/M \rceil \rceil$
- Costo = 2N * (# di passi)
- Quanto grande puo' essere il file da ordinare?
  - Con due passi: M*(M-1)
  - Con tre passi: $M*(M-1)^2$
  - Se la memoria e' di 100 MB e un blocco e' di 16Kb, M=6400
    - Due passi: $4*10^7$ blocchi=> 670 GB
    - Tre passi: $2,6*10^{11}$=>4,3 Petabytes! ($4,3 * 10^{15}$)
  - Con due passi si riescono a ordinare file molto grandi
    - Tempo di sort: 4N I/O di disco

Algoritmo di sort interno

- Quicksort e’ un modo veloce per ordinare in memoria centrale.
- Il tempo di sort interno si puo’ trascurare

Tempo di sort

- File di 100.000 blocchi (N)
- Memoria di 100MB
- Blocchi di 16.384 bytes
- =>M=6400, $\lceil 100.000/6400 \rceil = \lceil 15{,}625 \rceil = 16$=> riempiamo la memoria 16 volte, ordiniamo i record e scriviamo le liste ordinate su disco. L'ultima sottolista e' piu' corta, occupa solo 4000 blocchi.
- Secondo passo: merge delle 16 sottoliste

Tempo di sort

- Tempo di sort: 4N I/O di disco
- Se i blocchi sono disposti a caso sul disco, allora il tempo medio di accesso ad un blocco e' di 11ms
- Quindi tempo=100,000*4*11*10^{-3}=4400 secondi=74 minuti